MEMORANDUM
RM-4390-PR
FEBRUARY 1965

NATURAL LANGUAGE IN COMPUTER FORM

Martin Kay and Theodore Ziehe

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-700 monitored by the Directorate of Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

PREFACE

The shortage of text on magnetic tape, for both research on language and for operation of information-retrieval systems, has caused great difficulty in the past. But now, as a by-product of a pilot machine-translation project, the Air Force is producing a large amount of text, and print readers and automatic type-setting systems are almost ready to furnish more text than current facilities can accommodate.

Text on tape is an unstandardized commodity. Many schemes have been suggested for encoding and formatting information about source, type style, and other important features of text. As part of its continuing work in linguistics, The RAND Corporation--in collaboration with groups at the University of California, Berkeley, and the Faculté des Sciences, Grenoble--has developed a scheme that goes somewhat further in scope and flexibility than its predecessors. Several large collections of text are being put into the format described here. The computer programs being written at RAND to manipulate the format are available to others who may wish to avoid duplicating all or part of the work preliminary to linguistic research, literary scholarship, or information retrieval when computer support is required.

This Memorandum is intended for prospective users of the scheme and for designers of language processing systems.

It assumes little or no previous knowledge of computation or linguistics, and omits many details of programs--some of which remain to be worked out.

SUMMARY

This Memorandum describes a scheme for recording text in computer-usable form in such a way that all meaningful typographical distinctions are represented in a standard way. Provision is made for texts in different languages and different alphabets and for subsidiary material such as parallel translations and comments of interest to users and librarians. The basic set of encoding conventions is indefinitely extensible to accommodate new kinds of material.

Very large bodies of data require special facilities, and these have been provided by embedding the text encoding scheme in a general file maintenance system. This provides for a comprehensive set of labels for different-sized units of text and makes it easy to retrieve any given unit. It also provides means of correcting and revising material in the file.

It is expected that files of computer-usable text will be built up in a variety of ways and, in particular, that the keypunching of text for this express purpose will become steadily less important. Computer programs are described which simplify conversion of text from these various sources into the standard format.

The final section discusses the problem of printing text which has been recorded in the standard format and describes a flexible program for doing this.

ACKNOWLEDGMENTS

This Memorandum is the first product of a close collaborative effort among the Linguistic Project of The RAND Corporation, the Centre d'Etudes pour la Traduction Automatique of the Centre National de la Recherche Scientifique in Grenoble, and the Machine Translation Project at the University of California, Berkeley. The ideas expounded here came from many people in all three of these centers; only errors and misrepresentations should be ascribed uniquely to us. We are particularly indebted to Professor B. Vauquois and G. Veillon of Grenoble and to C. D. Johnson of Berkeley. D. G. Hays at The RAND Corporation has given us invaluable criticism and encouragement throughout the preparation of the manuscript.

CONTENTS

NATURAL LANGUAGE IN COMPUTER FORM

1. INTRODUCTION

The use of computers in linguistic and literary study is rapidly becoming more widespread and the natural aversion of humanists in general to mechanical methods is decreasing as the proper plaçe of machines in their pursuits becomes clearer. The amount of material that the student of language or literature must consider in the course of a single study is typically large and usually must be worked over many times.

The fortunate scholar can delegate much of this drudgery to students and research assistants; the yet more fortunate can hand it over to a machine. A linguist needing examples for a grammar, a lexicographer citations for a dictionary, a lawyer references for a brief, a preacher quotations for a sermon, can find them in concordances and indexes prepared by machine or may have them expressly sought by a computer. Words and phrases, rhymes and assonances, dactyls and spondees, aorists and pluperfects, can be docketed, counted, and compared, with never a comma missed. One manuscript can be compared with another and the differences classified; the works of several authors can be surveyed and evidence for the attribution of disputed works collected; bibliographies can be created, maintained and searched; sentences can be generated and parsed; all this and more can be done by machine if programs are available and if the texts are in a form that the machine can read. But there's the rub.

Programming is long and exacting work requiring much skill and experience. This is particularly true of the programming of non-numerical tasks, for programmers are rarely well trained in the appropriate techniques and the many aids and devices which are taken for granted in more traditional numerical computation are largely absent here. However, this situation is improving rapidly as a result of new programming languages and techniques that are being developed. Powerful and flexible programs for carrying out the kinds of operation needed by linguists and literary scholars will shortly be available in the libraries of major computing centers, just as those required by statisticians and physicists are now. A scientist is rightly dissatisfied if he must devote a significant proportion of his time to preparing computer programs. A humanist should be all the more so, since programming is clearly and properly alien to his background.

Collections of powerful linguistic programs are indeed necessary, but they can be of real value only if a satisfactory solution can be found to the problem of obtaining texts in a form the machine can read. This problem is in many ways more vexing than that of programming and is the subject of this manual. Hitherto, it has always been necessary to type out any text to be processed by a computer on a special machine which produces a coded copy of it on punched cards or on paper tape. This is particularly burdensome since the gains in speed and accuracy which come with the use of computers can all too easily be offset if large bodies of text have to be typed out by the researcher or an assistant, proofread, corrected, and modified.

There are several developments which may be expected to ease the heavy burden of typing and keypunching. The first machines capable of reading an ordinary printed page quickly and accurately are now available. There is great demand for them and they will rapidly become better, cheaper, and more widely used. Also, it is becoming clear that the business of producing books and periodicals will benefit greatly from mechanization. Some publishers are already preparing texts in machine form as the first step in their production process. Corrections, line adjustment, pagination, and the final typesetting are then carried out mechanically. The tapes or cards used in the final step of this process can be appropriated by research workers as input to computers.

As text coded on tapes and cards becomes more plentiful, libraries of such materials will doubtless be established, as they have been for microfilm, phonograph records, and the like. Each year, the chance will be greater that the text required by a particular scholar will already have been put into machine form.

This brings us face to face with the central concern of this manual--the question of standards for natural-language text in machine-readable form. As we have seen, computers can be expected to become part of the everyday equipment of linguists and literary scholars only if there is widespread cooperation among all concerned. Mechanical data processing can serve linguistic and literary studies as it ought only if programs and machine-readable text become as public as books, phonograph records, and films. However, to make a commodity public, it is not always sufficient to put it on the open market. A phonograph record which had to be played at 54 r.p.m. would not be really public; neither would an otherwise perfectly normal English newspaper if the letters on each line were arranged from right to left instead of from left to right. It is therefore important that there be general agreement upon the form in whi.' information is to be published.

Standard forms are already fairly well established for the grosser aspects of machine-readable material. The size and shape of punched cards are standard, and most manufacturers of computers, at least in the United States, use the same size of magnetic tape and the same kind of associated recording equipment. Most punched paper tape conforms to one of four standards, of which one is slowly gaining the ascendancy. Machines are, however, available which can be used to read paper tape of all four kinds.

On the other hand, there are not yet any standards for transcribing texts in ordinary language onto any of these media. How shall chapters, sections, and paragraphs be set off from one another? How are titles, subtitles, captions, and footnotes to be represented? What is to be done with scientific and mathematical formulae? How shall the distinctions among italics, boldface, and underlining be preserved? What of accented letters and unusual alphabets? These are, of course, questions which arise whenever we transcribe anything but the simplest kind of text on an ordinary typewriter, for an ordinary typewriter can make only a limited set of marks. But where the computer is concerned, we are denied the typist's ultimate resort of inserting exotic characters by hand. Furthermore, the conventions used for the computer must be carefully chosen and meticulously followed because the computer does not have the human facility for making inferences from context or for overlooking minor inconsistencies of convention. Good conventions are difficult to establish, and this is, in itself, good reason for having a standard, well-designed set which all may use.

But how can standard conventions for recording text be established when the physical form of the media used varies from place to place and from time to time? Some people use cards and some paper tape; some feed cards or tape directly into the computer whenever they wish to process it, whereas most first transfer material to magnetic tape or disk, which can be processed more rapidly by a large computer and is more easily stored. A limited amount of information can be punched on a single card, whereas there is much greater freedom with tape. This disparity is indeed troublesome, but less so than at first appears. If large libraries of machine-readable text are to be built up, the principal storage medium used must clearly be cheap, flexible, widely used, durable, compact, and of essentially unlimited capacity. Among the media available at present, magnetic tape best meets these criteria. Furthermore, as we shall see, the fact that text must be recorded on cards or paper tape before transfer to magnetic tape is an advantage rather than a drawback of this medium.

Consider the case of a man who is working on the New Testament and wishes to use a computer to perform certain operations on the Greek text. A general set of text-encoding conventions must clearly provide for texts which use the Greek alphabet, even though machines which have these symbols on their keyboards are rare. Normally, a transliteration scheme must be used at the keyboard and a computer program used to convert this into the standard format. The person who does the typing or keypunching can choose the transliteration which he finds easiest to use, and he need not concern himself with inserting marks to warn the machine that certain characters are to be interpreted in special ways. In any case, a program must be used to transfer the text from the medium on which it is originally punched onto magnetic tape, and this program can be made to do all the necessary conversions. The program can be provided with a table telling it that the text is in Greek and giving the transliteration scheme; it must also be told how chapter and verse divisions are being marked and various other clerical details of this kind. The important point is that the standard format need not be an additional burden to the typist or keypuncher. On the other hand, the designer of the standard format need not be constrained by a concern for the typist. A truly general set of coding conventions is unlikely to be easy to read and write, and it is therefore well that it should never need to be handled in its full complexity by anything but a computer program.

Printing text is, in many ways, similar to typing it. When a computer prints a text, or the results of some operations on a text, it must not be required to follow a single invariable set of rules. Here again, a program intervenes to prepare results in the form most suited to the user's needs, using his own conventions, transliterations, and page layout. Magnetic tape is just sufficiently remote from the everyday concerns of the user to provide him with the generality and flexibility he needs without troubling him with all the details. These details are, of course, given in full in this manual, but the reader should bear in mind that they need concern him only in the initial phases of a project. Once he has decided which of the facilities provided in the system he needs and has set up the necessary input and printing programs, he is free to work in his own way. In Sec. 4 and 5 we give examples of the kinds of typing and printing conventions that can be used and explain in greater detail how conversions are made between these and the standard form.

If a library of machine-readable text were set up for the benefit of a number of users or if one person had a particularly large amount of information to process, then the question of how to keep track of it all would soon arise. It would be necessary to divide the lines of text into units analogous to chapters, books, and collections of related books. Each of these divisions would have to be appropriately labeled so that it could be referred to easily and found when required; so that material could be changed, corrected, or modified; and so that other material could be put in the file before or after it. With such a system would come the need to store a new kind of information which would not be true text but rather information about the text--names of sections, acquisition dates, names of people who have modified or corrected a certain piece of text, indexing information, and so forth. All these things are provided for in the scheme described in this manual, in the form of a simple but powerful set of overall conventions called the catalog format; this is described in Sec. 3.

The reader who is not familiar with computers or programming will find that parts of this manual will not be clear to him on first reading. These parts concern material with which the linguist or literary scholar will need to concern himself only if he intends to become his own programmer--a course which we cannot recommend. This is a manual rather than a textbook and therefore contains much information for reference only. A programmer who sets to work to write basic reading and writing programs, updating and maintenance systems for libraries of text in the format proposed will, of course, have to work back and forth through these details time and time again. Much of this basic programming work has been done for a few computers,* and the programs are generally available.

*IBM 7040, 7044, and 7090.

2. CODES

A printed page contains letters of whatever sizes, colors, and shapes the typographer's art puts at the service of author, designer, and editor. A magnetic tape--as furnished, for example, by IBM--contains nothing but a very long sequence of binary digits (bits), ones and zeros, blocked in units of 6. The purpose of the somewhat complex codes described in the present section is to simplify the representation of textual matter in binary form. One should not attempt to keep in mind at every moment the configuration of 1's and 0's that represent, say, a Greek alpha or a mathematician's integral sign. Nor would it be convenient to specify a 10-, 11-, or 12-bit pattern for each of the thousand or more charactors that must somehow be represented.

2.1. Hollerith Characters

If we use six bits as the basic unit, there will be exactly 64 distinguishable patterns, and one of 64 characters can be recorded in each unit. Since it is inconvenient to write out 6-bit patterns in discussing machine operation, the customary scheme is to use octal digits to stand for 3-bit patterns. Table 1 shows the correspondence. Thus, the name of "000000" is "00", the name of "000001" is "01", and so on. The name of "111111" is "77". A reader who is not familiar with binary and octal numbers can best regard the 64 names assigned in this way as perfectly arbitrary.

Table 1

BINARY PATTERNS AND OCTAL DIGITS

Binary	Octal		Binary	Octal
000	0		100	4
001	1		101	5
010	2		110	6
011	3		111	7

The 64 characters can be copied from magnetic tape into the storage device of an electronic computer, and copied from storage onto tape. Only 48 of them can normally be printed on paper for a human reader; for many years, printers controlled by punched cards were designed to print 48 different marks, and the tradition has continued. The 48 standard marks include the letters of the Roman alphabet, the ten decimal digits, a few marks of punctuation, arithmetic signs, and a blank. The other 16 characters that can be recorded on magnetic tape but are not normally printed are designated here by their numbers. Table 2 associates the binary patterns, the octal equivalents, and the 48 "Hollerith" characters that we have been discussing.

Table 2

BINARY PATTERNS, OCTAL EQUIVALENTS, AND HOLLERITH MARKS

B	O	H	B	O	H	B	O	H	B	O	H	B	O	H
000 000	00	0	001 101	15		011 010	32		100 111	47	P	110 100	64	U
000 001	01	1	001 110	16		011 011	33	.	101 000	50	Q	110 101	65	V
000 010	02	2	001 111	17		011 100	34	)	101 001	51	R	110 110	66	W
000 011	03		010 000	20	+	011 101	35		101 010	52		110 111	67	X
000 100	04	4	010 001	21	A	011 110	36		101 011	53	$	111 000	70	Y
000 101	05	5	010 010	22	B	011 111	37		101 100	54	*	111 001	71	Z
000 110	06	6	010 011	23	C	100 000	40	-	101 101	55		111 010	72	
000 111	07	7	010 100	24	D	100 001	41	J	101 110	56		111 011	73	,
001 000	10	8	010 101	25	E	100 010	42	K	101 111	57		111 100	74	(
001 001	11	9	010 110	26	F	100 011	43	L	110 000	60[a]		111 101	75	
001 010	12		010 111	27	G	100 100	44	M	110 001	61	/	111 110	76	
001 011	13	=	011 000	30	H	100 101	45	N	110 010	62	S	111 111	77	
001 100	14	'	011 001	31	I	100 110	46	O	110 011	63	T			

[a]Octal 60 corresponds to the Hollerith blank.

Tradition is strong, but modern necessities are stronger. The 48 Hollerith characters are no longer the only ones that can be printed. Other sets of 48, sets of 120, and even sets of 240 are now obtainable. In fact, direct output to machines with all the flexibility of Linotype or Monotype machines is possible. This very powerful printing capability provides added incentive to develop a flexible encoding scheme. We return to questions of output below.

Since the design of most computing machines compels us to work with 64 primitive characters, we have two feasible alternatives for the recording of ordinary texts in which a much larger character set is used. We may represent each distinct printer's mark with a sequence of two or more recordable characters or we may adopt the functional equivalent of the case shift on a typewriter. When the shift lock on the typewriter is depressed, the effect of striking a key is altered. Each key is associated with two marks; one mark is obtained when the machine is in its lower-case shift, the other when it is in upper case. The number of shifts can be increased beyond two, and their use can be extended, for example, to the differentiation of Greek and Roman letters.

2.2. The Roman Alphabet

The 64 recordable characters are distributed, by standard machines, into a set of 48 that can be printed and a set of 16 that cannot. We take 15 of the nonprintable characters as alphabet flags. The flag for the Roman alphabet is (octal) 35 = (binary) 011101. After an occurrence of this flag, all following characters on a tape are taken to represent letters of the Roman alphabet until another alphabet flag occurs to signal a shift into a different alphabet.

The Roman alphabet is intended for use in transcribing English text, as well as French, German, Portuguese, Rumanian, and others. The 26 letters of the English alphabet are therefore supplemented by ten diacritics. Most punctuation is excluded from the Roman alphabet, to be kept in another (a "punctuation" alphabet), but one mark, the apostrophe, has been retained.

Five characters are used to identify special fonts of type. These identifiers can also be thought of as shift indicators. They can be used singly or in combination to modify the characters in the current alphabet. The absence of any font identifier is understood to mean lower case, standard font. An identifier is reserved for upper case, another for italics or script, a third for bold face, one for larger type than ordinary, and one for smaller type. Strung together, they can identify words or phrases in several font combinations; e.g., bold-face italics, all caps small size, large-size italics. Three characters are used to identify sequences of marks that are printed as superscripts or subscripts with respect to the ordinary line or with letter spacing in the European manner. Still another character is used to identify the end of any one or combination of the shifts described here.

To complete the set of 48 recordable characters, the Roman alphabet includes a space and an unassigned character.

As examples of use of shift indicators within the Roman alphabet, consider the following encoding; the first line is from source text, the second is Hollerith representation of the characters on magnetic tape:

We seek the coöperation of all men of good will.

```
1W9E SEEK THE CO-OPERATION OF ALL MEN OF GOOD WILL
```

C'était à l'époque de NAPOLÉON que le héro est né.

```
1C9'(ETAIT )A L'(EPOQUE DE 1NAPOL(EON9 QUE LE H(ERO
   EST N(E
```

Smith^{a} and Jones^{b} have disagreed violently.

```
1S9MITH6A9 AND 1J9ONES6B9 HAVE 2DISAGREED9 VIOLENTLY
```

In these examples, terminal periods have been omitted from the Hollerith representations. Punctuation belongs in another alphabet and would require flags, use of which is demonstrated in Sec. 2.4. Table 3 recapitulates the coding of the Roman alphabet.

Table 3

CODE: ROMAN ALPHABET[a]

Letters									Diacritics			Shifts		
T	O	H	T	O	H	T	O	H	T	O	H	T	O	H
A	21	A	J	41	J	S	62	S	ó	74	(	Upper case	01	1
B	22	B	K	42	K	T	63	T	ò	34	)	Italics or script	02	2
C	23	C	L	43	L	U	64	U	ö	13	=	Bold face	03	3
D	24	D	M	44	M	V	65	V	o̧	73	,	Larger size	04	4
E	25	E	N	45	N	W	66	W	ô	20	+	Smaller size	05	5
F	26	F	O	46	O	X	67	X	õ	53	$	Superscript	06	6
G	27	G	P	47	P	Y	70	Y	ǒ	54	*	Subscript	07	7
H	30	H	Q	50	Q	Z	71	Z	ȯ	33	.	Letter spacing	10	8
I	31	I	R	51	R	'	14	'	ō	40	-	Shift terminator	11	9
									ø	61	/	Blank	60	

[a]Column headings: T = text, O = octal, H = Hollerith. In German text, ß is encoded as 73-62, i.e., S with cedilla. In the presentation of diacritics in this table, the letter o is used only to provide a base; the diacritic is encoded independently of the letter it accompanies, as on a typewriter with dead keys.

2.3 The Cyrillic Alphabet

The Cyrillic alphabet, used for transcription of Russian text, includes 32 letters. Since no diacritics are needed, there is no difficulty about fitting this alphabet into a set of 48 recordable characters. In addition to the Cyrillic letters, there are represented an apostrophe, a blank, the shift indicators used in the Roman alphabet, and the terminator needed to identify the end of any shift or combination of shifts. Five characters are left unassigned. The code is given in Table 4.

Table 4

CODE: CYRILLIC ALPHABET[a]

T	O	H	X	T	O	H	X	T	O	H	X	T	O	H	X
А	14	'	A	И	31	I	J	Т	50	Q	T	Ы	67	X	Y
Б	21	A	B	К	34	)	K	У	51	R	U	Ь	70	Y	'
В	22	B	V	Л	41	J	L	Ф	53	$	F	Э	71	Z	,
Г	23	C	G	М	42	K	M	Х	54	*	X	Ю	73	,	(
Д	24	D	D	Н	43	L	N	Ц	62	S	C	Я	74	(	=
Е	25	E	E	О	44	M	O	Ч	63	T	H	'	13	=	)
Ж	26	F	*	П	45	N	P	Ш	64	U	W				
З	27	G	Z	Р	46	O	R	Щ	65	V	Q				
И	30	H	I	С	47	P	S	Ъ	66	W	$				

[a]Column headings: T = text, O = octal, H = Hollerith, X = transliteration. Shifts and blank are treated exactly as in the Roman alphabet (see Table 3). No Cyrillic diacritics are provided.

In the following example, the first line is from source text, the second is Hollerith representation of the characters on magnetic tape, and the third is Hollerith-character transliteration of the tape records. Again, alphabet flags are not shown.

```
Пучок протонов из электростатического генератора
1N9RTM) NOMQMLMB HG ZJE)QOMPQ'QHTEP)MCM CELEO'QMO'
1P9UHOK PROTONOV IZ ,LEKTROSTATIHESKOGO GENERATORA
```

2.4. Three Special Alphabets

The Greek alphabet has been coded primarily to permit transcription of mathematical texts and others in which Greek letters are used to denote special concepts. Thus, although an alphabet flag has been assigned and Hollerith characters specified for recording the Greek letters on tape, the characters printed are not uniformly satisfactory as a translation. See Table 5 for this code.

The variety of special marks used in commercial and scientific text is so large that it is necessary to define a full set of Hollerith characters for them. The arabic numerals, signs for dollars and cents, arithmetic symbols, and so on, have been given code characters (see Table 6), with 17 more characters available.

The following example illustrates the use of the symbol alphabet. Alphabet flags are shown as underlined octal integers.

Table 5

CODE: GREEK ALPHABET[a]

Letters																Diacritics			
T	O	H	X	T	O	H	X	T	O	H	X	T	O	H	X	T	O	H	X
Α	21	A	A	Η	27	G	Y	Ν	44	M	N	Τ	62	S	T	ό	74	(	(
Β	22	B	B	Θ	30	H	V	Ξ	45	N	C	Υ	63	T	U	ὸ	34	)	)
Γ	23	C	G	Ι	31	I	I	Ο	46	O	O	Φ	64	U	F	ὀ	14	'	'
Δ	24	D	D	Κ	41	J	K	Π	47	P	P	Χ	65	V	X	ὁ	53	$	H
Ε	25	E	E	Λ	42	K	L	Ρ	50	Q	R	Ψ	66	W	Q	ô	20	+	+
Ζ	26	F	Z	Μ	43	L	M	Σ	51	R	S	Ω	67	X	W	ŏ	13	=	=

[a]Column headings: T = text, O = octal, H = Hollerith, X = transliteration. Shifts and blanks are treated exactly as in the Roman alphabet (see Table 3).

Table 6

CODE: SYMBOLS[a]

Numerals			Arithmetic Symbols			Shifts		
T	O	H	T	O	H	T	O	H
0	00	0	$	53	$	Italic or script	62	S
1	01	1	*	54	*	Bold face	63	T
2	02	2	¢	23	C	Larger size	64	U
3	03	3	&	21	A	Smaller size	65	V
4	04	4	=	13	=	Super-script	66	W
5	05	5	+	20	+	Subscript	67	X
6	06	6	-	40	-	Character spacing	70	Y
7	07	7	×	44	M	Shift terminator	71	Z
8	10	8	÷	24	D			
9	11	9	.	33	.			
			,	73	,			
			§	47	P			
			Space	60				

[a]Column headings: T = text, O = octal, H = Hollerith. The period and comma in this alphabet are used as a decimal point and a digit position marker respectively and not as ordinary punctuation.

They paid $17,000 for them, 85% of $20 thousand.

<u>35</u>1T9HEY PAID <u>55</u>$17,000 <u>35</u>FOR THEM<u>15</u>, <u>55</u>85% <u>35</u>OF <u>55</u>$20 <u>35</u>THOUSAND<u>15</u>.

Here the underscored numbers are alphabet flags: 35 for Roman; 55 for symbols; 15 for punctuation.

Some builders of text files will use the Greek and symbol alphabets to transcribe precisely the symbolic material in their sources. Others will prefer to shorten their labors by using cover symbols to replace complex expressions. They may feel, nevertheless, that it is important to differentiate classes of complex expressions according to their ranges of syntactic functions. In transcribing Russian text at RAND, we adopted this procedure and defined an alphabet of cover symbols. Table 7 describes this alphabet, and an editor's manual* describes use of it. Others who use cover symbols will presumably need to establish categories appropriate to their own purposes.

2.5. Punctuation and Boundaries

An alphabet of punctuation marks is included in the present system to allow unambiguous representation on tape of the many marks that occur in text. Table 8 lists the marks for which characters have been assigned; 17 additional marks can be added to the alphabet at will.

---

*Edmundson, H. P., <u>et al.</u>, <u>Studies in Machine Translation - 4: Manual for Pre-editing Russian Scientific Text</u>, The RAND Corporation, RM-2065-1, revised June 1961.

Table 7

CODE: COVER SYMBOLS[a]

Text	Octal	Hollerith
Fractions, dates, Roman numerals	05	5
Degrees	0501	51
Degrees of latitude or longitude	050101	511
Hours and minutes (when written with subscripts or superscripts)	050102	512
Percentages	0502	52
Numerals preceded by ordinal sign ("N$^{\underline{o}}$", "#", etc.)	0503	53
Numerals followed by single letters	050301	531
Bibliography - reference numbers	0504	54
Cardinal numbers ending with "1" except "11"	0505	55
Cardinal numbers ending with "2" "3", or "4", except "12", "13", and "14"	0506	56
Cardinal numbers ending with "0" or "5" through "9", and the numbers "11" through "19"	0507	57
Ordinary technical expressions	06	6
Chemical expressions	0601	61
Table (when part of sentence)	0603	63
Reference asterisk	0604	64
Relational symbols used alone	0605	65
Ordinary relations	07	7
Chemical relations	0701	71

[continued on following page]

Table 7 - Continued

Text	Octal	Hollerith
Ratios	0702	72
Half relations	0705	75
Chemical half relations	0706	76
Plural indicator	64	U
Space	60	

[a]These symbols were designed for recording articles from some Russian scientific journals. This accounts for the classification of numerals by last digit. A symbol code followed by the plural marker indicates a sequence of symbols of that type. This alphabet is indefinitely extensible as other applications arise.

Table 8

CODE: PUNCTUATION[a]

Text		Octal	Hollerith
(	Open parenthesis	74	(
)	Close parenthesis	34	)
"	Open quotation mark (double)	21	A
"	Close quotation mark (double)	22	B
'	Open quotation mark (single)	23	C
'	Close quotation mark (single)	24	D
[	Open brackets	25	E
]	Close brackets	26	F
«	Open double angles	27	G
»	Close double angles	30	H
.	Period	33	.
:	Colon	31	I
;	Semicolon	41	J
,	Comma	73	,
-	Hyphon	40	-
—	Dash	47	P
?	Question mark	42	K
¿	Open-question mark (Spanish)	43	L
!	Exclamation mark	44	M
¡	Open exclamation (Spanish)	45	N
...	Ellipsis	46	O
/	Slash	61	/
	Space	60	

[a]Shift codes are the same as in the Roman alphabet (see Table 3), except that no upper-case shift is provided.

Two linguistic boundaries are given special status for the benefit of editors who choose to mark sentences and paragraphs in their text files. These two boundaries are represented by two characters in a boundary alphabet (Table 9) which is otherwise free for uses yet to be defined.

Table 9

CODE: BOUNDARIES

Text	Octal	Hollerith
Beginning of paragraph	47	P
Beginning of sentence	62	S
Space	60	

2.6. Flags, Shifts, and Fillers

Some of the 64 characters that can be recorded on tape are used to represent printed marks; the others are either flags or shifts. The flag characters are not said to belong to any alphabet; the shift characters belong to every alphabet, although they have no obvious use in the alphabet of cover symbols. This difference reflects the fact that shifts are part of the linguistic description of a text, whereas flags are only a computational device.

Flags are defined because our tape machines record and recognize 6-bit patterns. Only flag characters have independent definitions; the meaning of a flag is always the same, regardless of context. Each flag signifies that the following nonflag, or text, characters are to be interpreted by reference to a certain alphabet. Table 10 shows the flag codes and the alphabets that they designate.

Table 10

CODE: ALPHABET FLAGS

Alphabet	Octal
Roman	35
Cyrillic	36
Greek	37
Symbols	55
Cover symbols	56
Punctuation	15
Boundaries	16
—	—
Filler	77

The shifts that we define are graphemes. It is true that their use could be avoided if we had a sufficiently large set of recordable characters, but using them has advantages beyond mere reduction of the size of the character set. For example, consultation of a dictionary is more convenient if the same word, printed in normal type in one place and in bold face elsewhere, is spelled with the same characters in both places. Isolation of a _bold face_ grapheme gives us this advantage without losing the information about type style.

The shift graphemes could have been defined in several ways. Thus, the _range_ of a shift could extend (i) over a single letter, (ii) over all letters up to the following blank, (iii) over all letters up to the next shift, or the next shift in a given class, (iv) over all letters up to a terminator specific to the given shift, or (v) over all letters up to a generalized terminator. In our choice of the fifth plan, three criteria of simplicity were relevant: size of alphabet, length of text, and complexity of rules. Plans (i) and (ii) would save one character, the terminator, but lengthen the text; and plan (ii) would not permit encoding of, for example, "_un_happiness". Plan (iii) would require introduction of a _normal_ shift, perhaps several, in place of the terminator. Plan (iv) would call for more terminators; it would tend to shorten text, as when a word is italicized within a bold-face line, but such occurrences are rare.

It is important to understand that shifts can be used in combination. For example, we have already noted that bold-face italic can be represented by two shift indicators preceding a string of characters, but it is also possible to use a string of identical shifts to represent an extreme degree of an attribute. Thus,

X6Y6Z

represents x^{y^z}, and

X6Y7Z

represents an occurrence of x with a superscript composed of y with subscripted z. Of course, if the occurrence of z were back on the principal line, the encoded representation would be

X6Y9Z

In similar fashion,

LOOK

encodes a word in normal-size type,

4LOOK

encodes a word in large type,

44LOOK

encodes a word in still larger type, and so on. Naturally, the editor of a text will not attempt to distinguish more sizes than are essential to retain the meaning.

Of the 64 characters recordable on tape, only one remains to be explained; it serves as a filler. The 6-bit patterns on magnetic tape are recorded and read in groups of 6. Naturally, it is not always possible to code a segment of text so that the number of significant characters is divisible by 6, nor would it be advisable to use blanks, which might have an unintended significance. Hence octal 77, with no Hollerith equivalent, is set aside to serve as a filler. To illustrate use of the filler: If a segment of text should require 63 characters for recording on tape, there would be 10 groups of 6 characters each, then one group of 3 characters plus 3 fillers.

3. ORGANIZED FILES OF TEXT

The user of a file of text on tape will sometimes take the largest available quantity and handle all of it uniformly, collecting, without regard to source, all the instances of a phenomenon that interests him. For other purposes, he will need to discriminate in some way; he may choose to examine only introductory paragraphs in scientific articles or only articles on a certain subject or only dialogue in plays and novels or only titles. To serve him well, text files must be organized and labeled. In Sec. 2 we described the tape representation of unorganized units of text, e.g., printed lines; here we go on to the representation of organizational features--conventions for assembly of many lines of text in a Chinese-box arrangement that we call catalog format.

Instead of proposing special formats only for text, we apply conventions that are equally appropriate for dictionaries, grammars, and other sorts of material. We thus save the cost of writing many primitive computer programs to synthesize and analyze statements about form; independent of the substance of the catalog, such programs deal _only_ with form. If it were necessary to write new programs for files of different materials, we would be tempted to simplify the organization of each file. Since the catalog formats can be used for every file, we feel free to elaborate the organization of each file as far as profitable.

3.1. Maps

A catalog is a collection of _data_. Each datum can be large or small, but the datum is the smallest manipulable unit for the programs that ope.ate on catalogs as catalogs; the content of the data is the substance of the catalog. In a text file, most data contain representations of text in the code given in Sec. 2; these data are called _text entries_ and each can usually be assumed to contain a line of text. The organization of the file is accomplished by data called _labels_, each marking a significant group of entries. These labels are nested at four levels: A collection of entries is called a _section_ and labeled; a labeled collection of sections is a _division_; a _corpus_ label is applied to a collection of divisions; a _text file_ is a sequence of corpora and has no label. Any entry or labeled group of entries can be annotated in _description_ data.

The catalog-management system uses diagrams like the one in Fig. 1 to specify the form of a catalog. The diagram is a _tree_; at its origin is the label for a corpus

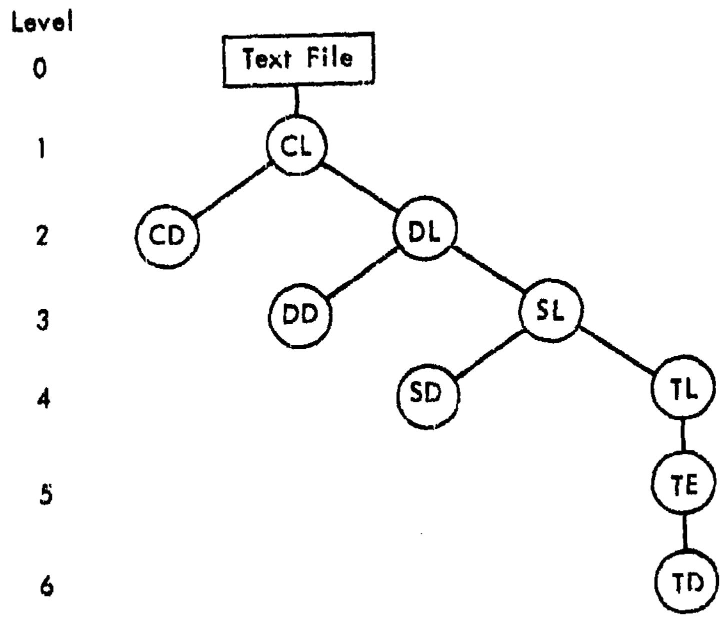


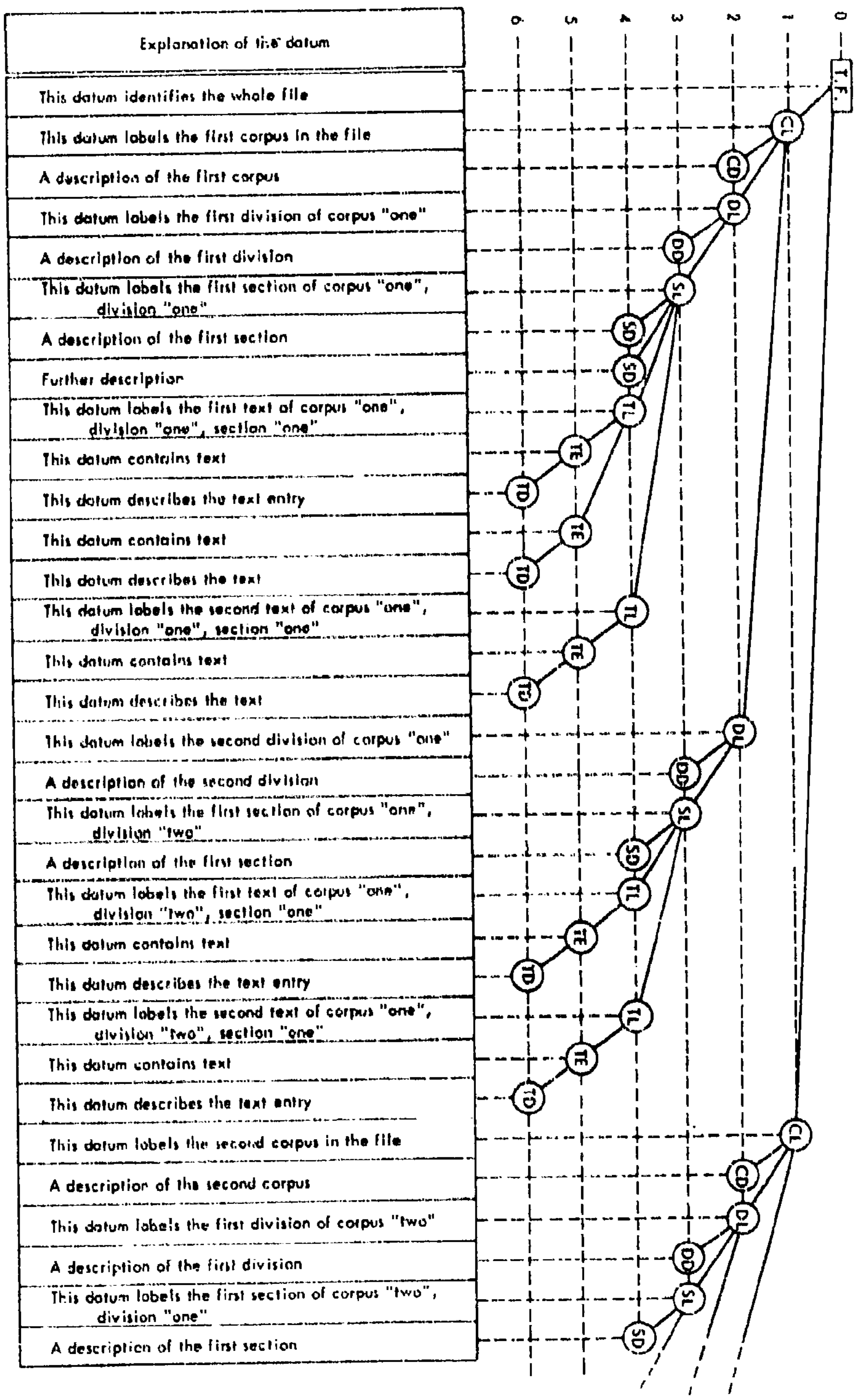
Fig. 1—Map of a text file in catalog format
(see text for explanation of symbols)

(CL = corpus label). Under this node, on the second level, are two data-class names, CD = corpus description and DL = division label. In these remarkable maps, a single data-class name stands for an unlimited number of data of the named class, just as if one airport on a map of the world could stand for all the world's airports. This simplification, which enables us to draw one map for all catalogs of text before even the first is constructed, is feasible because the catalog-management system imposes enough conventions to guarantee that every datum can be found when the map is followed.

To complete our inspection of Fig. 1, let us use the map as a guide to the preparation of a magnetic tape. We can record only one thing at a time as the tape unrolls; we use the map to decide what datum to take at each step. Starting at the top, we record a corpus label. Next we go to level 2 and begin at the left with a corpus description; if several data of this class are to be recorded, we take them one after another. When all description data for the corpus are on the tape, we move to the next node on the same level, which in this map is a division label. We record the first division label for the corpus; perhaps we intend to record several divisions, each with a label, but we must take everything in the first division, that is, data of all classes named under the DL node in the map, before recording a label for the second division. Thus the map requires that when we have recorded the first division label we move at once to level 3 and record division descriptions (DD). Since there are no nodes under DD, we take as many data of this class as we require and then proceed across level 3 to node SL. Recording the first section label, we move down to level 4 and record our first text label (TL), down to level 5 for a text entry (TE), and down to level 6 for a text description (TD).

Now we are at the bottom of the map. The next datum to be recorded is another text description for the same entry, if any; otherwise, another text entry with its text descriptions under the same text label; otherwise, the next text label within the section. When the last text label in a section has been recorded, followed by its text entries and their descriptions, another section label goes on the tape. What follows the second and each subsequent section label is just like the material following the first, with, of course, no restriction on the number of text labels in any section. When everything pertaining to the last section in a division has been recorded, a new division is begun with its label, description data, and so on. When everything pertaining to the last division is on the tape, a new corpus is begun. The end of the last corpus is the end of the catalog.

In Fig. 2 we present an expanded map with one node for each datum, and a diagram descriptive of material entered on a



Explanation of the datum
This datum identifies the whole file
This datum labels the first corpus in the file
A description of the first corpus
This datum labels the first division of corpus "one"
A description of the first division
This datum labels the first section of corpus "one", division "one"
A description of the first section
Further description
This datum labels the first text of corpus "one", division "one", section "one"
This datum contains text
This datum describes the text entry
This datum contains text
This datum describes the text
This datum labels the second text of corpus "one", division "one", section "one"
This datum contains text
This datum describes the text
This datum labels the second division of corpus "one"
A description of the second division
This datum labels the first section of corpus "one", division "two"
A description of the first section
This datum labels the first text of corpus "one", division "two", section "one"
This datum contains text
This datum describes the text entry
This datum labels the second text of corpus "one", division "two", section "one"
This datum contains text
This datum describes the text entry
This datum labels the second corpus in the file
A description of the second corpus
This datum labels the first division of corpus "two"
A description of the first division
This datum labels the first section of corpus "two", division "one"
A description of the first section

Fig. 2—The tree structure of a sample catalog

corresponding hypothetical tape. The figure is to be read like a diagram of syntactic dependencies; each node governs those on the level just below and connected to it, and each node is associated with a segment of the tape diagram by a dotted line.

The main outlines of the map for text files are obvious. A corpus is an enormous block of text; for example, all the Russian text prepared for machine processing at RAND is recorded as two corpora. A division in this corpus corresponds to an issue of a journal, to a book, or to some similar publication unit. Each section is an article or chapter. Descriptions are optional and contain whatever information a particular user feels to be necessary. The description of a corpus can tell in what language it is written, what center produced it, and any other information which would be useful at this size level. The description of a division can specify a scientific discipline, give bibliographic information, and the like. The description of a section can include, for example, the page numbers of the original publication. Text labels are used to differentiate titles, subtitles, authors, summaries, footnotes, body, and the like. They also identify page and line in a way that makes checking the source convenient. Text descriptions, on the other hand, are used for annotation of a freer kind. The map shows that several entry data can follow a text label; this device permits recording of interlinear texts, in which the first entry under a label contains an original-language line, the second a translation, and so on. Since text descriptions are connected to text entries, not to text labels, the original-language line can be described in one datum, the translation in another.

The catalog-management system permits null data. Although the map shows a text-description datum under each text entry, this facility would not be equally convenient for all users of the format. It would be wasteful to require every user to insert descriptions, wanted or not, for the sake of those whose text must be described. A null datum occupies no space on tape; it need not be mentioned during preparation of a tape file, and it does not appear when the file is consulted. Thanks to this technique, programs for manipulation of text need not be altered when descriptions are added or deleted; descriptions can be added to any file without alteration of the map (which is the same for every text file) or conversion of programs; and a library containing some text with descriptions and some without can be scanned from end to end without special attention to this variation.

Data of any class can be left null. For example, it would be possible to record data for several divisions within a single corpus without recording any division labels; the results might be bad, but the boundaries between divisions would be discernible nevertheless.

3.2. The Data of a Text File

Form and content are clearly distinguished by the catalog-management system. In Sec. 3.3 we describe the way the system uses data to control form; here we examine the several classes of data in which the substance and organization of a text file are recorded. They are the classes named in the map, Fig. 1.

Each _text entry_ is a string of characters, encoded in accordance with the rules of Sec. 2. Since each entry is independent of all others, each must begin with a flag character.

Description data of all types are encoded as text; Sec. 2 applies to them also. It is thus possible to write descriptions with all the elegance of the printer's art, while at the same time a single complex of programs will suffice for the decoding of all data in a text file.

The labels are written with Hollerith characters in restricted formats. Let us take them class by class.

Corpus labels. Each datum in this class consists of 72 characters. The first 24 constitute a corpus name; the remainder are for notes. As many of these characters as desired can be left blank (not null!). If descriptions are for scholarly use, notes within labels are for use by programmers and librarians. Here the manager of a text file can indicate what corrections have been made to a corpus, what special procedures have been applied, and the like.

Division labels. The length of the datum is again 72 characters, but only two are allowed for the name of the division, all the rest being given over to notes. The purpose of the restriction on name length is to encourage a brevity that will be gratifying to all who must write division names repeatedly in instructions for correction, scanning, or analysis of text. The notes can be used for an expanded version of the name, or for librarian's records.

Section labels. The same format is used as for the division labels, and with the same argument.

Text labels. It is intended that a text label be written for each line of original printed text. Since a file will contain a great many text labels, the length of each is limited to six characters. The first character is a type symbol, the remainder an entry name. The type symbols, listed in Table 11, are redundant in a certain sense; they could presumably be reconstructed from the spaces and shift symbols contained within the entry. Since the program for the reconstruction job would be complex, and since users may often want to exclude titles, authors' names, etc., from a search, the redundant symbols have been inserted.

The entry name included in the text label is large in order to permit organization below the level of the section. For example, the five characters can be used for page number and line number, or for scene and line number, or in whatever manner is natural to the text being recorded.

Table 11

TEXT TYPE SYMBOLS

T - Title	A - Author
S - Summary	E - Editor's Note
M - Major Section Heading	I - Intermediate Section Heading
N - Minor Section Heading	B - Body
F - Footnote	L - Bibliography ("Literature")
D - Stage Direction	P - Speaker (Drama)

3.3. Control of Form

The structure of every catalog is described within the catalog itself by means of control terms. We need only skotch these terms and their functions here, since the catalog-management system deserves separate treatment. Indeed, what we have to say about them is of interest mainly to programmers rather than to linguists or others who want to record text in the system. The programs described in Sec. 4 make use of control data in reading catalog tapes and construct them when data are recorded, but in many applications it will not be necessary to do more than take the programs as written and put them to work.

A catalog is recorded on one or more reels of magnetic tape. We estimate that a 2400-foot reel of tape recorded at IBM's highest density will store about 1.5 million running words of text without descriptions, but some large text files will require more than one reel. As we have said, information is recorded on tape as a sequence of 6-bit characters, and these characters are read and recorded six at a time. A <u>physical record</u> is a sequence of 36-bit spans preceded and followed by blank tape. Since the space left blank between physical records is long enough for the storage of nearly 600 characters, it is good economy to make records long. On the other hand, a complete physical record must be read or recorded without pause; hence programmers must always set aside enough space in machine memory to contain a record, and memory space is a precious commodity. Our compromise is to limit physical records to 2400 characters, including control data, but this figure is arbitrary and limits may be set to meet needs of users.

The first physical record of each catalog is a <u>control record</u>, and the last is an <u>end-of-file</u> mark. The end-of-file mark may be followed by another control word beginning a new catalog unrelated in any systematic way to the first. On the other hand, it may contain other information recorded in an altogether different format. The point is that the catalog structures are confined within the boundaries set by end-of-file marks.

Two control data are stored in the control record at the beginning of each tape. The first is a management datum, the second a map datum.

<u>The management datum</u>. This datum consists of exactly four 36-bit spans, used as follows:

(1) Record size limit. The number of 36-bit spans ("words") allowed in one physical record on this tape, written as a binary number.

(2) Date. The date of writing of the catalog, recorded in Hollerith, with two characters each for month, day, and year.

(3) Reel number. The sequence number of the reel of tape within the text file, recorded as a binary number.

(4) Repeated data. The number of content data repeated (from preceding reels) at the start of this one.

The fourth item is needed because a user may wish to use a single reel out of a multi-reel text file. When he does, he must identify the first new datum on the selected reel as standing at a certain node in the map of the catalog. To enable him to do so, the necessary data are repeated from reel to reel, but the number repeated is not constant. Item (4) shows where the new material on the reel begins.

<u>The map datum</u>. The symbols in the map of the catalog are listed here in a definite order. A symbol is recorded before those below it in the map, and those below it before those to the right. Thirty-six bits are used to describe each data-class:

(1) Data-class name. Three Hollerith characters, or 18 bits.

(2) Code. The set of rules to be used in encoding or decoding a datum of this class is identified by one Hollerith character. B = Binary, R = Regular (or RAND; the code of Sec. 2 above), or others to be defined.

(3) Empty. Three bits not presently used.

(4) Level. The level number of this data class in the map, recorded in binary. 9 bits.

Since the map is given at the beginning of each tape reel, it is possible to use any reel out of a file without special arrangements.

We now turn to the assembly of data into physical records. Each datum is required to fit the Procrustean bed of machine memory, with its 36-bit cells. We describe the control terms added to records as prefixes and suffixes; when they are taken into account, the physical record also fills a whole number of cells.

The <u>datum control prefix</u> occupies 54 bits, as follows:

(1) Empty. The first 3 bits contain binary zeroes.

(2) Datum length. The number of 36-bit cells filled by this datum is given as a 15-bit binary number.

(3) Empty. The next 9 bits contain binary zeroes.

(4) Data-class name. The class of this datum is identified by its line number in the map at the

beginning of the tape reel, given as a 9-bit binary number.

(5) Preceding implicit level (PIL). This is defined as follows: (i) The PIL of all data on level 1 is 0; (ii) The PIL of the <u>first</u> datum dominated by a null datum is the PIL of the dominating null datum; (iii) The PIL of every other datum is the level of the datum that dominates it. This is given as a 9-bit binary number.

(6) Own level. The map level of this datum, given as a 9-bit binary number.

These control terms allow generalized catalog programs to find boundaries of all levels in a catalog tape, no matter how many data in whatever combinations are null. The length of the datum is used in determining where the next prefix begins.

The <u>datum control suffix</u> occupies 18 bits, as follows:

(1) End of catalog. A single bit, 1 if this is the last datum in the catalog and zero otherwise.

(2) End of tape. A single bit to mark the end of a reel.

(3) End of physical record. A single bit.

(4) Datum length. Same as item (2) of the datum control prefix.

The end markers have an obvious use in programs that read catalogs. The datum length is given in both prefix and suffix in order to facilitate reading from end of tape to beginning as well as in the normal direction.

The data, with their prefixes and suffixes, are put together, with a record control prefix and suffix to complete a physical record.

The rocord control prefix occupies 90 bits, as follows:

(1) IOBS tag. Three bits, 101, to allow use of the IOBS level of the IBM Input-Output Control System.

(2) Record length. The number of machine memory cells occupied by the record, less one, recorded as a 15-bit binary number.

(3) IOBS tag. Three bits, 010.

(4) Empty. Nine bits, all zeroes.

(5) IOBS tag. The letter M, Hollerith, six bits.

(6) FORTRAN tag. Three binary zeroes.

(7) Record length. The number of machine memory cells occupied by the record, less two, recorded as a 15-bit binary number.

(8) Empty. Seventeen binary zeroes followed by a binary 1.

(9) Start of catalog. A single bit, 1 if this is the first physical record after the first control record of a catalog and zero otherwise.

(10) Start of tape. A single bit to mark the start of a reel.

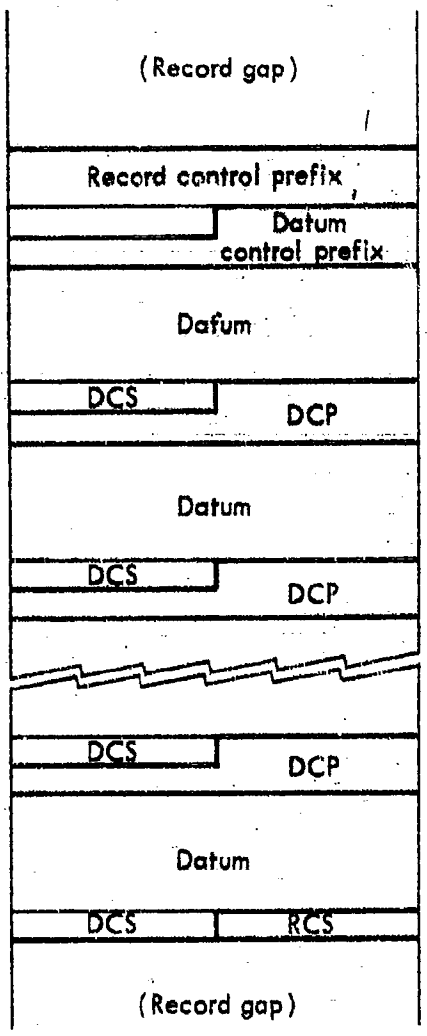
(11) Start of physical record. Always a binary one.

(12) Empty. Fifteen bits, all zeroes.

Items (1) through (5), occupying one memory cell, supply the IOBS program with information it requires. Items (6) through (8) do the same for the FORTRAN input program. Neither of these programs is used by the catalog-management system, but provision for their use is relatively inexpensive and possibly advantageous to some future user of the text. Items (9) through (11) serve as end marks for the catalog system when it reads tape from end to beginning.

The record control suffix is eighteen bits, all zero.

Figure 3 shows how the data, with their prefixes and suffixes, fit together to form a physical record. In every record there are four 36-bit cells ahead of the first datum, two between every pair of adjacent data, and one at the end. This layout is followed for every record, including the control record at the beginning of a tape. Only the end-of-file mark is different.



DCS:	Datum control suffix
DCP:	Datum control prefix
RCS:	Record control suffix
Record gap:	Blank space between consecutive physical records

Fig.3—Layout of a physical record on tape

4. WRITING TEXT CATALOGS

We have observed that the standard catalog format will rarely, if ever, be the first machine-readable form given to a text. Recording on magnetic tape will be from punched sources, usually cards or paper tape. It is therefore inevitable that at least part of any computer program used for putting text into the standard catalog format be written especially for the kind of punched source used. Scholars working with texts in different languages and for different purposes will require their own computer routines, though these need differ from the standard routines only in small details. Routines for converting teletypesetter tape and the tapes used in printing books will differ more markedly from standard. Very special routines are required to convert Monotype tape for use as computer input; there are almost as many conventions for recording text on Monotype tape as there are for all the other media together. In this section, we shall see how computer programs can be designed to convert punched text from a wide variety of sources into the catalog format, the necessary changes being confined to a very restricted part of the program.

A program to put punched text into catalog format must be able to do two things: It must be able to substitute for the character codes of the original machine-readable text the proper sequences of flags, shifts, and character codes in the standard catalog format; and it must be able to recognize the boundaries of entries, sections, divisions and corpora of text and to label them correctly.

An investigator who punches all of his own text on cards or tape can arrange to incorporate the labels he needs at the appropriate places in the text. But if the original coded material (e.g., teletypesetter tape) was not prepared with computer usage in mind, labels will have to be punched separately and merged with the text by the routine which sets up the catalog. The routine must therefore be able to accept input from at least two sources at once. If the new text is to be inserted into, or added at the end of, an existing catalog, then the existing catalog must be read by the routine, making a possible three inputs at the same time.

New text, existing catalogs, and labels constitute the three principal _input channels_ of the program to be described. There are also two auxiliary input channels: one to provide a set of _pointers_ which establish the relationship between new text entries arriving on one input channel and a set of catalog labels arriving on another; and the other to provide corpus, division, section, and text descriptions. We shall describe how these various input channels are used in the following pages.

The text writer program must be prepared to accept data in different formats from each of its different input channels. Existing catalogs will, of course, always be in the same format. Labels and pointers will be prepared by a given investigator and can readily be made to fit a standard format. Descriptions will usually also be prepared by the investigator and there should be a standard method of punching them. However, descriptions are so similar to ordinary text entries that text-entry flexibility probably should be extended to them. The text writing program has, therefore, a central controlling section which writes the catalog. Data are passed to this controlling section by four more or less standard peripheral sections which read and convert existing catalogs, labels, pointers, and descriptions, and a fifth peripheral section responsible for reading and converting new text. This last section of the program, called the keypunched text reader (KTR), is the only part which has to be changed to accommodate different kinds of text.

4.1. A KTR for Punched Cards

The simplest application of the text writer program is one in which a text, specially keypunched on cards or paper tape to be put into catalog format, is to be converted and put on magnetic tape. In such an application, the keypunching conventions can be designed to be as convenient as possible. All necessary labels and descriptions can be punched along with the main body of the text so that only one of the input channels of the text writer program will be required.

Let us assume that punched cards are to be used as the input medium. Punched cards have the property that each of the 80 columns represents one character; there are no back-space or underline codes which could result in the number of columns and character positions getting out of step. This makes it convenient to distinguish certain kinds of information by the positions on the card where they are punched. In this case, we shall set aside the first five columns of each card for a simple serial number so that, should the cards be dropped or mixed up in some other way, they can be reordered mechanically. Five columns suffice for 100,000 different numbers which is more than we are ever likely to need. However, if the cards are numbered in tens (00010, 00020, 00030...) it will always be possible to insert other cards in case some are accidentally omitted in the first punching or in case a correction results in two cards where there was originally one. The KTR section of the text writing program will check that these serial numbers appear in nondecreasing sequence. Since it will not check that the sequence is strictly ascending, the sure footed and those with small amounts of text to punch may leave these five columns blank. In any case, sequence numbers can be inserted automatically either before or after the other columns of the cards have been punched.

Column 6 of each card will be left blank for the moment except on the very last card of the deck which will be marked with an "E" for "end." This column may later be used for correcting and updating the text once it has been put on tape.

Column 7 will show the kind of information the card carries. Corpus, division, and section labels will be represented by the letters "U", "V", and "W", respectively; their descriptions will be represented by "X", "Y", and "Z". Since text labels are short and occur frequently, it would be inconvenient and wasteful of cards to punch each one on a separate card. We shall therefore arrange to punch text labels on the same cards as the entries themselves. However, there are instances where two or more text entries are covered by the same text label and we must be sure that the entries are properly distinguished on the cards. With this in mind, we shall put a "1" in column 7 for the first entry under a given label, a "2" for the second, a "3" for the third, and so on. If the KTR finds two cards with the same text label and the same number in column 7, it will assume that the second continues the entry begun in the first; if the labels are the same but the number in column 7 is one higher than on the previous card, it will assume that a new entry under the same label has begun. The ability to put more than one entry under the same label will normally be used only for interlinear text and therefore it will create no hardship to limit to nine the number of such entries in this format.

Each entry of text may be accompanied by a description. The description of the first text entry under a given text label will carry the code "A" in column 7, that of the second "B" and so on. In this particular input format, each text-description card is assumed to be a separate entry. With these conventions, there is nothing to prevent a description being provided for a text entry which does not exist. This is less eccentric than it sounds because, if we were to use the same card arrangement for making corrections to the file, it might be necessary to insert or change a description while leaving unaltered the entry described.

The remaining 73 columns will depend on the kind of information carried on the card. A corpus-label card, coded "U" in column 7, will have the 24-character corpus name in columns 8 through 31 and the corpus notes in columns 32 through 79. The first card (00010) shown in the example (Fig. 4) is a corpus-label card; the name of the corpus is "MOLIERE'S PLAYS" and the notes remind the user that this text was put on tape on 10/2/64 and that the tape was corrected on 10/17/64. Card 00020 in the example is a corpus-description card. A corpus-description card has an "X" in column 7, columns 8 through 13 are left blank, and the description itself begins in column 14 and may run to the end of the card; it is punched using the same character-encoding conventions as for the body of the text. Columns 8 through 13 are left blank in order to

```
00010 UMOLIERE'S PLAYS.       10/2/64 CORRECTED 10/17/64
00020 X      3OXFORD 3EDITION.
00030 VP1
00040 WA1
00050 1TB000104LE BOURGEOIS GENTILHOMME4.
00060 1MB000204ACT ONE4.
00070 1IB000304SCENE ONE4.
00080 1DB000403L'OUVERTURE SE FAIT PAR UN GRAND ASSEMBLAGE
00090 1DB0005D'INSTRUMENTS+ ET DANS LE MILIEU DU TH5E7ATRE ON
00100 1DB0006VOIT UN 5EL6EVE DU 3MA7ITRE DE MUSIQUE, QUI COMPOSE
00110 1DB0007SUR UNE TABLE UN AIR QUE LE 3BOURGEOIS A DEMAND5E
00120 1DB0008POUR UNE S5ER5ENADE.
00130 1PB000904MA7ITRE DE MUSIQUE4.
00140 1DB0010PARLANT 6A SES MUSICIENS.
00150 1BB001103VENEZ, ENTREZ DANS CETTE SALLE, ET VOUS REPOSEZ L6A EN
00160 1BB0012ATTENDANT QU'IL VIENNE.
00170 1PB001304MA7ITRE 6A DANSER4.
00180 1DB0014PARLANT AUX DANSEURS.
00190 1BB001503ET VOUS AUSSI, DE CE C7OT5E.
00200 1PB001604MA7ITRE DE MUSIQUE4.
00210 1DB00176A L5EL6EVE.
00220 1BB001803EST-CE FAIT$
      .
      .
      .
      .
```

Fig. 4 - Example of Punched-card Text Format

make description cards as nearly as possible like ordinary entry cards, which will have a label in this space.

A division name is composed of only two characters, and these are punched in columns 8 and 9, immediately following the "V" in column 7. The remaining columns of the card are available for notes. In Fig. 4 (card 00030), the beginning of one division, called "P1", is shown. There are no notes associated with it. Section labels are similar to division labels--two characters of section name in columns 8 and 9 followed by notes. Text entries and their associated descriptions are provided with six-character text labels which are punched in columns 8 through 13. The first character of a text-entry label declares the type of the entry (see Sec. 3, Table 11) and the other five characters are an arbitrary name. There are text entries of six types shown in Fig. 4. Card 00050 carries a title and has a "T" as the first character of its label, i.e., in column 8. Card 00060 gives the act in the play which is regarded as a "major section heading" (coded "M" in column 8). Scene headings (as card 00070) are marked "I" for "intermediate section heading." Each speaker is announced in an entry with column 8 tag "P", and "D" is used for the stage directions. The remaining text entries constitute the body of the text and are marked with a "B" in column 8. There is, of course, no obligation to punch the title at all, or it may be put more out of the way as description. If it is included in the ordinary text entries, it will always be treated as text by the computer. If it is included as descriptive material, it can more easily be passed over when the text is processed. However, this is largely a matter of taste. Figure 5 summarizes this card layout.

In the example given in Fig. 4, we have assumed a corpus consisting of Molière's plays. Each division might be a different play and each section a different act or scene. Nothing of importance turns on these decisions, which are on a par with the decision as to whether an act should always start at the head of a new page in the book.

So much for the card layout. We must now take up the question of how the text itself is to be encoded. Of all the characters available in the various alphabets (see Sec. 2) of this catalog system, only a few can be expected to occur in any given text, and even fewer with any frequency. Furthermore, not all the frequently occurring characters can be expected to belong to the same code alphabet. For example, characters from the punctuation and, possibly, the boundary alphabets will occur fairly frequently in English texts, whereas the diacritics of the Roman alphabet will be fairly rare. The most desirable solution would be to allow the keypuncher freedom to assign each of the 48 Hollerith characters to a symbol which occurs frequently in the text he happens to be working on. These assignments

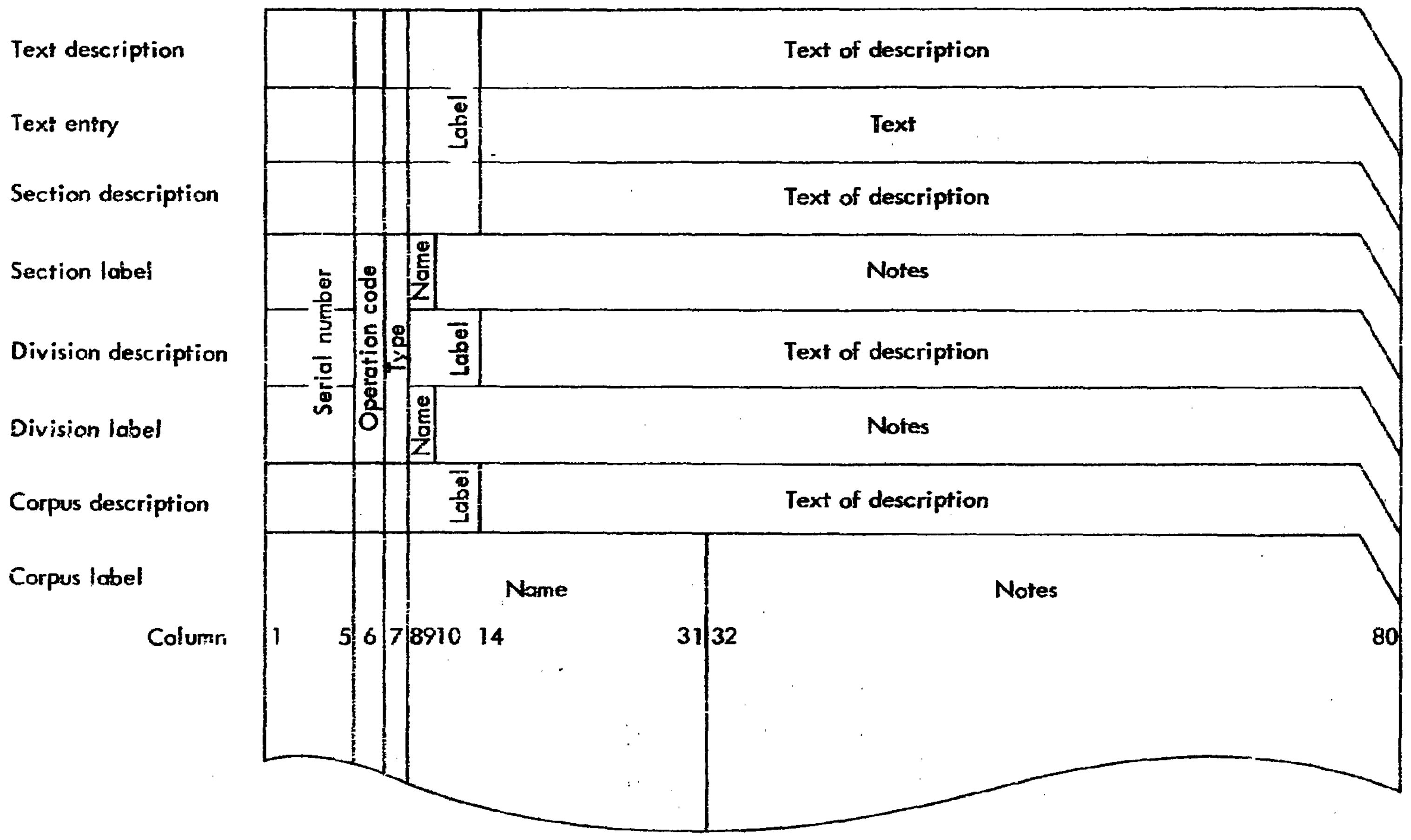


Fig. 5 — A punched-card input layout

would be taken to hold except when some special device was used to indicate a departure from them. If 47 of the Hollerith characters were assigned to frequent symbols, then the 48th could be used as this special device; otherwise one of the 16 codes which have no corresponding Hollerith character could be used for this purpose.

In the example in Fig. 4, the text was in French so that five of the diacritics are needed. The cedilla occurs only with "c" and the two can therefore be combined for keypunching purposes. The diaresis is not frequent in French, but it is tidy to include it with the other accents and there is room for it among the 48 characters available. In many texts, the beginnings of sentences will not be marked; however, they are marked here if only to make the example more instructive. Capital letters are used for proper names and at the beginnings of sentences, and a simple way of showing capitalization is required. Also, there are occasional stretches of text, such as the headings of the speeches, which are all in capitals. Where a word begins with a capital, there is no real necessity to use both an up and a down shift; one character placed before the affected letter should be sufficient. On the other hand, the use of this character before each of the letters of a capitalized word would be cumbersome. The solution is to have two methods of indicating capitalization. One Hollerith character, "3" in our example, is used for "short" capitalization;

it affects only the immediately following letter and, therefore, requires no down shift. Another character, "4", is used for "long" capitalization; its effect continues until cancelled. It is not necessary to commit a third character for cancelling long capitalization; the same character, "4", can be used again in this function without ambiguity.

Table 12, which shows KTR Table 1, gives most of the assignments of text characters to Hollerith characters for keypunching the French in the example, the remainder are given in Table 13, which shows KTR Table 2. The characters found in the original text are given in the column headed "T", and the characters punched into the cards are given in the column headed "P". The remaining two columns must now be explained.

We have made some liberal assumptions about the KTR routine we are describing. If the keypuncher is to be left free to assign interpretations to the Hollerith characters as he will, he must also have some means of conveying to the KTR routine what assignments he has, in fact, made. He can do this in only one way; namely, by punching some other cards which the KTR routine will read before it encounters the text and which contain the necessary correspondences. The columns headed "P", "N", and "I" contain all the information that need be punched into these cards. Consider the first entry in Table 12. Column P tells us

Table 12

TABLE ONE OF A PUNCHED-CARD KTR

T	P	N	I	T	P	N	I	T	P	N	I
a	A	1	RA	A	3A	1	R1RAR9	.	.	1	P.
b	B	1	RB	B	3B	1	R1RBR9	,	,	1	P,
c	C	1	RC	C	3C	1	R1RCR9	;	+	1	PJ
d	D	1	RD	D	3D	1	R1RDR9	:	=	1	PI
e	E	1	RE	E	3E	1	R1RER9	!	/	1	PM
f	F	1	RF	F	3F	1	R1RFR9	?	$	1	PK
g	G	1	RG	G	3G	1	R1RGR9	-	-	1	P-
h	H	1	RH	H	3H	1	R1RHR9	'	'	1	R'
i	I	1	RI	I	3I	1	R1RIR9	(	(	1	P(
j	J	1	RJ	J	3J	1	R1RJR9	)	)	1	P)
k	K	1	RK	K	3K	1	R1RKR9	Open "	1	1	PA
l	L	1	RL	L	3L	1	R1RLR9	Close "	2	1	PB
m	M	1	RM	M	3M	1	R1RMR9	Space		1	R
n	N	1	RN	N	3N	1	R1RNR9	Long cap.	4	2	R1

[continued on next page]

Table 12 - Continued

T	P	N	I	T	P	N	I	T	P	N	I
o	Ø	1	RØ	Ø	3Ø	1	R1RØR9	´	5	1	R(
p	P	1	RP	P	3P	1	R1RPR9	`	6	1	R)
q	Q	1	RQ	Q	3Q	1	R1RQR9	^	7	1	R+
r	R	1	RR	R	3R	1	R1RRR9	¯	8	1	R=
s	S	1	RS	S	3S	1	R1RSR9	ç	9	1	R,PC
t	T	1	RT	T	3T	1	R1RTR9	Rm. Alph.	*R	R	
u	U	1	RU	U	3U	1	R1RUR9	Cy. Alph.	*C	C	
v	V	1	RV	V	3V	1	R1RVR9	Gk. Alph.	*G	G	
w	W	1	RW	W	3W	1	R1RWR9	Sy. Alph.	*A	A	
x	X	1	RX	X	3X	1	R1RXR9	Cs. Alph.	*D	D	
y	Y	1	RY	Y	3Y	1	R1RYR9	Pu. Alph.	*P	P	
z	Z	1	RZ	Z	3Z	1	R1RZR9	By. Alph.	*B	B	
Sentence	0	1	BS								

Table 13

TABLE TWO OF A PUNCHED-CARD KTR

T	P	N	I	T	P	N	I	T	P	N	I
A	A	2	RA	N	N	2	RN	´	5	2	R(
B	B	2	RB	Ø	Ø	2	RØ	`	6	2	R)
C	C	2	RC	P	P	2	RP	^	7	2	R+
D	D	2	RD	Q	Q	2	RQ	¨	8	2	R=
E	E	2	RE	R	R	2	RR	ç	9	2	R,RC
F	F	2	RF	S	S	2	RS	Space		2	R
G	G	2	RG	T	T	2	RT	Sentence	0	2	BSR1
H	H	2	RH	U	U	2	RU	End cap.	4	1	R9
I	I	2	RI	V	V	2	RV				
J	J	2	RJ	W	W	2	RW				
K	K	2	RK	X	X	2	RX				
L	L	2	RL	Y	Y	2	RY				
M	M	2	RM	Z	Z	2	RZ				

that an "a" in text is to be punched as "A". Column N says that the next character in the text is also to be interpreted by reference to this table (which will be KTR Table 1 from the computer's point of view.) The "RA" in column I indicates exactly what must be put on tape; namely, the character represented by a Hollerith "A" in the Roman alphabet (see Table 3 for Roman alphabet). The "ç" is to be punched as "9" and, once again, the following character in text is to be interpreted by reference to this same table. The representation on tape is shown as "R,RC", that is, the Roman alphabet flag, if necessary, the character in the Roman alphabet represented by a Hollerith "," followed by the character in the Roman alphabet represented by a Hollerith "C".

Take, now, the case of a capitalized word or sequence of words. Such a sequence is preceded by a "4" on the punched card, which is the code adopted for long capitalization. This is a very special kind of code because it influences the interpretation of an indefinitely long sequence of codes which follow it. Thus, the punched "4" signals a complete change in the table used for interpreting the succeeding punched characters. Thus, in Table 12, opposite "Long cap." in column T, we find a "4" in column P and a "2" in column N. This "2" means that KTR Table 2 is to be used for interpreting punched characters from this point until further notice. KTR Table 2 is similar to

KTR Table 1 except that it allows rather few codes. For example, we have taken the view, however arbitrarily, that all punctuation is lower case; it therefore does not appear in KTR Table 2. The code for "beginning of sentence" is in the boundary alphabet (see Table 9), and we are allowing for the possibility of a beginning of sentence in a capitalized sequence. In order to do this, we must first record the beginning-of-sentence code with the appropriate alphabet flag, and then return to upper-case shift in the Roman alphabet. This accounts for the sequence of four characters in column I. Finally, the character punched as "4" has an entirely different interpretation in this table. It now means "shift into lower case." Accordingly, it causes the shift terminator, "9" in the Roman alphabet, to be recorded on tape, and indicates that KTR Table 1 is once again to be used for interpreting succeeding punched characters.

It will now be clear that in both KTR tables the characters in column I come in pairs; the first of each pair specifies an alphabet flag and the second a character in that alphabet. If nothing appears in this column, then nothing will be recorded on tape; if an alphabet flag appears with no character following it, a blank will be recorded on tape. Thus we are able to specify all the characters in all the code alphabets and to distinguish the blank from total absence of a character while using only the 48 symbols that can be keypunched. It need not be supposed that, because we show an alphabet flag before each substantive character in column I, these will all be written on tape. The program will ensure that only necessary alphabet flags and shifts are actually recorded.

One important facility of this KTR still remains to be explained. What is to be done if, from time to time, a character must be punched which, although it can be represented perfectly easily in the tape format, is not provided in the tables constructed for the job on hand. For this purpose, the punch codes "*R", "*C", "*G", etc., have been provided. "*G", for example, causes the Greek alphabet built into the system to be used directly; the Greek letters are represented by the Hollerith codes given for them in Table 5. If it was required to punch a tilde, which is in the Roman alphabet in the system but not in the alphabet we have been using for the French example, it would be sufficient to punch "*R$" (see Table 3). The codes "*G", or "*R", etc., must be repeated before every character to which they apply.

The KTR routine we have been discussing is one which will be implemented on the IBM 7044 computer. It is designed to allow great flexibility, but, as always, this comes at a certain cost. A simpler routine would confine its user to a more rigid set of coding conventions, but would avoid the necessity of setting up detailed tables.

On the other hand, it is undoubtedly a great deal simpler to construct tables of the kind we have described than to write, or have written, a special KTR routine for each particular job. Clearly other routines of equal or greater flexibility could, and doubtless will, be written for punched-card as well as other input media.

4.2. Punched Paper Tape

In the KTR which has been described, two methods have been used for distinguishing different kinds of information, card layout and character codes. To a considerable extent, it is possible to barter these against one another. If a serial number is to be useful for sorting punched cards into order when they get into disarray, it must be always in the same columns, but all the other information could have been punched anywhere on the cards and distinguished by character codes. "$E" might have been used to separate entries, "$S" sections, and so forth. The first six characters following an entry separator might be understood to be a text label. Conventions of this kind are usually best avoided with punched cards because distinctions based on position on the card are particularly easy for both humans and computers to make, and because the character set is already severely enough restricted without pre-empting other character combinations unnecessarily.

The method of distinguishing types of information by position is also simple to use with paper tape. In this case, positions are referred to not by column number but by counting character positions from the last line-feed. A character position is not necessarily the same thing as a row of holes on the tape itself, and in this respect, paper tape is different from punched cards. Character positions get out of step with paper-tape codes in a way they never can with card columns. This arises in three ways: (i) When a wrong character is punched in paper tape, it is converted by the typist into a "delete code" which is ignored by all machines which process the tape, including the computer. (ii) When the shift key is depressed or released, a code is punched into the tape which, however, does not account for a character position. (iii) Paper-tape machines are sometimes equipped with a back-space key which causes a code to be punched. This gives it the capability of returning to character positions for which codes have already been issued and either issuing new codes by overtyping the old symbol or modifying what is there by underlining or adding diacritics.

The result of all this is that a given line of type can be produced in a variety of ways, each resulting in a different sequence of codes on the paper tape. A word can be underlined by first typing the whole word, then back-spacing over it and typing in the underline. The underline can be typed first and then the word. Each letter can be typed and underlined before going on to the next. The number of variations is endless, but the final result is the same. It is impossible to tell by inspecting the typescript if the typist spent some time between a certain pair of characters depressing and releasing the shift key; yet the paper tape records all these moves. A well designed KTR--or any other computer program--will reconstruct on the evidence of the punched codes the line of type on the hard copy. This it can do by "imagining" where the carriage of the typewriter would be placed as a result of each move. This reconstructed line would then serve as input to the KTR proper and could be treated in every way like the punched card. With this device, it is possible to make corrections by simply back-spacing over the offending characters and typeing corrections on top. Underlining and the insertion of diacritics can be done in any way that suits the typist short of actually man-handling the carriage, a recourse which, since it is essentially hidden from the computer, can only confuse it.

We have mentioned that punched paper tape takes various forms and comes from various sources; in particular, it is produced as a by-product of certain printing processes. A tape of this kind requires a special KTR to handle the special codes and conventions used for different fonts, type styles, etc. But there are also other problems which arise. An investigator who gets access to these tapes typically finds himself confronted with a large cardboard box into which the tape has been allowed to fall after running through the printing machine; but for the investigator, the box and its contents would have been thrown away. A KTR is written, and each length of tape is put on magnetic tape as, say, a separate section in a text catalog. Since the original order of the tapes is not known, they are taken in any order and the sections are given arbitrary names--sequence numbers will do. Now the tape is printed using a standard text-catalog printing program. We assume that the investigator has a hard-copy version of the text at hand. Using this, he can prepare a new set of labels for the sections which will be in ascending numerical (or alphabetical) order when the text is properly ordered. He now has the text writer program read the text catalog from one of its channels and the new labels from another. The original labels are replaced by the new ones. A standard sort program can now be used to put the sections of text in order using the new labels as keys. This may be the final form of the text catalog, or it may be necessary to re-label a second time if the investigator requires a particular set of labels which will not be in ascending order in the final catalog. This is one of the many uses to which a multi-channel text writing program can be put; others include merging catalogs, revising and correcting catalogs, extracting examples of particular kinds of occurrence from a text and writing them as a separate catalog and adding descriptions. Separate manuals will describe the details of how these operations can be performed with particular programs.

5. PRINTING

Text in the standard format must be printed from time to time to be proofread and corrected, to verify the intermediate results of a process, or to display final results; it must be printed if only to provide the investigator or librarian with an exact record of what he has on the tapes. As with the other processes we have mentioned, printing requires a computer program, and the particular programs used will vary considerably with the purposes to be achieved by printing. Either the emphasis is on showing character-for-character what is on the magnetic tape, or it is on producing a presentable copy for publication or circulation among colleagues.

It may not be immediately apparent why there should be these two points of view on printing. Surely, if real distinctions are made in a text which are worthy of preservation on magnetic tape, they must be made again when the text is reproduced; otherwise they are otiose. However, texts, or selections from texts, are printed for differing purposes and in different styles. If a list of the different words in a text is prepared, possibly with frequencies or other incidental information, the distinction between words with initial capital and those entirely in lower case is unlikely to be interesting; what is required is a standard style: all capitals, all lower case, or all with initial capital. As with capitals, so with italics, bold face, and other distinctions of type font. Consider also the case where citations are being printed to exemplify certain words or phrases, say one sentence for each citation. Here, upper and lower case should probably be preserved, but, for ease of reading, it may be convenient to capitalize or underline the cited forms.

Another reason for considering different printing conventions comes from the equipment generally available for printing. It is indeed the case that the output of a computer can be made indistinguishable from that of a high-class printing shop. It is not, as many believe, necessary for computer results to look like cold, squarely mechanical telegrams from a remote electronic god. Very few will be able to pay the price of special equipment capable of printing a very large budget of symbols in a large variety of sizes and type faces, and even those who can will find it uneconomical for anything but the final results of a major enterprise. All intermediate, and most final, results will continue for many years to be produced with more modest machinery.

The most common computer printing devices are capable of printing the same 48 Hollerith characters that appear on the keyboard of the card punch. A relatively simple modification to some of these, notably the IBM 1403 printer, enables them to print somewhat enlarged character sets; 120 or even 240 are possible on this machine at some cost in running speed. It is usually a trivial matter to replace the 48 standard characters on an ordinary machine by any other set of 48, and the characters of the enlarged sets can be chosen freely. However, neither of these expedients comes near to providing the flexibility that we have allowed in the text encoding scheme. It was, of course, worth allowing this flexibility against a possible eventual use of more powerful printing machines and, what is more important, against sophisticated machine processes which can take advantage of all the distinctions in the original text.

The problem, then, is similar to that encountered in keypunching text on a machine with only a small set of characters. Sometimes the difficulty can be overcome by dropping certain distinctions, as for example when a vocabulary list is printed all in capitals and without punctuation or diacritics. This can be useful for many purposes. But cases also commonly arise where everything on the tape must be represented on the printed copy. An obvious one is proofreading. If a correction is made on the hard copy produced for proofreading, this correction must be keypunched and presented to a program which will modify the tape. Clearly, the hard copy must contain everything the keypuncher must know in order correctly to reconstruct the catalog entry. In other words, the copy must be such that it would be possible to keypunch the text from it and arrive at exactly the same text catalog on tape as was produced from the original.

5.1. A Standard Printing Scheme

Many of the devices used for keypunching can conveniently be taken over to the printing process. The characters which occur most frequently in the text are each assigned one of the symbols on the printing device. The correspondence set up in this way is assumed to hold except when overridden by a special signal. A special signal takes the form of an alphabet flag or shift character which modifies the interpretation of all following symbols up to another special symbol. The special symbols can be distinguished from other symbols more clearly in printing than in keypunching if two lines of print are made to stand for one in the original text. The substantive text characters appear on alternate lines so that the spaces above them are always blank. An alphabet flag or shift is represented by a character on the upper line with a space on the line immediately below. Suppose that the following line has been encoded in the standard format and written on tape:

The Greek μήτηρ comes from the same root as the
    Russian Матъ.

This might be printed on a machine equipped with only the 48 Hollerith characters as:

```
1 9   1 9     G       G
 T HE  G REEK  M(YTYR  COMES FROM THE SAME ROOT AS

          1 9       C1 9   C
      THE  R USSIAN   M AT'
```

Using a machine with a large character set including, say, upper and lower case Roman lotters and accents, it might appear as:

```
           G      G
 The Greek  mýtyr  comes from the same root as the

         -    C    C
      Russian  Mat'
```

The first two words, "The Greek," are in the Roman alphabet, and no special symbol is needed to show how they are supposed to be interpreted. Most letters appear in lower case, and it is therefore understood that the upper case letters printed by the 48-character device should be taken as representing lower case except when preceded by the shift character "1" on the upper line. Upper and lower case shift characters are not printed by the device with the larger character set because it is able to show these distinctions directly. The Greek word "μήτηρ" appears in transliteration in both cases because we are assuming that neither printer is equipped with Greek characters. The fact that it is a transliteration, and from what alphabet, is shown by the two G's on the upper line which serve as a pair of brackets round the transliterated sequence. The same device is used for the Russian word "Матъ ".

This example would have been different if the original had read:

> The Greek mḗtēr comes from the same root as the
> Russian mat'

With the 48-character printer, this would appear as

```
1 9   1 9
 T HE  G REEK M-(ET-ER COMES FROM THE SAME ROOT AS

           1 9       '
       THE  R USSIAN MAT'
```

Here the transliterations are part of the original; no alphabet flags are needed because nothing but Roman letters and diacritics are used. The output of the second printer would be

> The Greek mē´tēr comes from the same root as the
> Russian mat'

In this example, we are making the realistic assumption that the device will be capable of printing only one diacritic over a letter; others must precede or follow.

The conventions required for printing a Russian or a Greek text would clearly be different. The best solution would be to use a printer with Russian or Greek characters and to transliterate symbols from other alphabets when they occurred. When this cannot be done, a more usual printer can be used with a different set of standard conventions about how characters are to be interpreted in the absence of alphabet flags. The Roman characters would normally be understood to be transliterations except when explicit "R" appeared on the upper line.

In most cases, a single investigator will work only with one kind of text so that, once he has established his conventions for keypunching and printing, he will not need to change them again. However, in a large center where there are many different kinds of text, or where several individuals are using the services of the same keypunch operators or proofreaders, some care must be taken to see that the various sets of conventions do not become confused. In these circumstances it is desirable that a printing routine be capable of printing as part of each batch of output, what conventions it is using. In other words, it should be able to read from cards or some other input medium tables similar to those used to specify keypunching conventions (Tables 12 and 13), to use these in preparing the printed output, and to print them on that output so that the intepretation will always be clear.

5.2. Page Layout

We have consistently taken the line that repetitive operations should, wherever possible, be carried out by standard computer routines whose internal workings need not concern the individual user. Printing is an obvious example of such an operation and, accordingly, we have sketched some features of a general printing program. This is the program used on the IBM 7044 computer at RAND. The principal routines involved can be used as subroutines in any program which produces printed output and not only to prepare a copy of a connected text. They are therefore provided with facilities for arranging information on the page in the way the user requires, for furnishing page headings and numbers and the like.

The printing routines prepare output in units called rows. A row may comprise several lines of print, though the number will usually not be very large. In general, a row is divided into columns whose number and width on the printed page is specified by the program using the routines, that is, by the user. In many instances, for example when a straightforward text is being printed, a row will consist of a single line of print and there will be a single column which fills the entire page. If each line is identified with a label, then there would be a small column for the label number, and a large column occupying the rest of the page for the text itself.

The number and width of columns may vary from row to row. Each variant is specified by a <u>format statement</u> which is nothing more than a sequence of numbers giving the widths of succeeding columns. Before the preparation of a row of print begins, the printing routines must be supplied with the format statement to be used for that row. Data are then supplied piecemeal, each with an indication of the column into which it is to be put and the tables to be used for transliterating it. It is thus possible to print, say, Greek, in one column according to one set of conventions and English in another according to another set.

The data in a given column may be accumulated in various ways. Suppose that a bilingual dictionary is being printed; we may assume that it has been compiled by examining some quantity of parallel text. Let one language be Russian and the other English. There will be two fairly wide columns to accommodate Russian and English words and phrases. The transliteration tables will be different for each of these. Two more columns will accommodate some grammatical information about the words and phrases; this will be a code of a few letters giving the part of speech. Finally, there will be a column for recording the number of times each Russian-English pair has been found in the text. An entry in the dictionary will have information in all of these columns and will be one row of print. The data for the various columns may be supplied by the user's program in any order and we may return to a column to which information has already been supplied as often as we wish. Russian words will begin at the left-most margin of their column and extend as far to the right as necessary; the rest of the column will be left blank. From time to time, a phrase will occur which is too long to fit in the column. It must therefore be broken at the last space which can be fitted on the line and the remainder used to begin a new line. This will continue until the whole phrase is accommodated. Entries in the English column will be handled in the same way. A given Russian word or phrase may have more than one English correspondent, and these must be set off from one another. To provide for this, the printing routines allow the program which uses them to specify, when providing data for a column, that it <u>must</u> begin on a new line. The user is, of course, also free to supply a line of blanks to separate items.

A Russian or English word or phrase may, let us suppose, have several part-of-speech codes and these are to appear under one another in a column. In this case, the printer must be instructed to go to a new line, not after the last space it can accommodate on the current line, but at every space. This is one of the options provided by the routines. The column giving the number of times a Russian-English pair has been found in text must also have no more than one item on a line. However, in this case the items are all numbers and we shall want to follow the convention of justifying them to the right so that digits with the same place value fall under one another. Therefore, when these data are sent to the printing routines, it is with an instruction to place them, one on a line, at the right of the field.

There are other instructions which may accompany data to the printer. In the examples we have considered, a long item is split up into as many lines as are needed to accommodate it in the field it is directed to. Sometimes, however, we may wish to print an item only if it will fit on the current line and otherwise to preserve it for a subsequent row or discard it entirely. When a simple copy of a text is being produced, for instance, we may wish to fit as many words on a line as will fit and keep the rest for subsequent lines. This is exactly what would happen if the entire text were printed as a single row with the instruction to go to a new line whenever all the characters up to the next space cannot be fitted on the current one. However, there are practical limits to the size of a single row which come from the fact that it occupies space in the working area of the computer while it is being assembled. It will rarely be practical to print rows representing a whole page of print for this reason. The facility of interrupting the flow of information to the printer when the end of a line has been reached is therefore provided.

Each time a row is printed, the paper advances a certain number of lines. This number is, in fact, the greatest number of lines filled in any column in the row. Unoccupied lines in the other columns are left blank and the next row begins with the next completely free line on the paper. Having printed a row, the user, or his program, may need to know how much space remains on the current page in order that he may decide whether to go to the next before printing the following row, and for this reason he is supplied with a running count of the place he has reached on the page.

When he goes to the next page, he will often want to start by printing a heading. Headings are like other rows of print except that they typically use their own format statement and the information they contain is largely the same from one use to the next. The page count will be different on each occasion, and a title or subtitle may change from time to time, but changes will be relatively rare. It may also happen in the body of a page that the information in a certain column will remain unchanged for a number of rows. To meet this situation, the printing routines can be so arranged that the information printed with a given format statement will always be as on the previous occasion except in those columns where it has been explicitly changed. Thus, in the case of a page heading, the column where the page number is printed will be changed each time the format statement is used whereas the remaining information will normally not be changed.

The printing routines we have described were designed for great flexibility both in the kinds of text they can handle and the printing devices they can serve. However, we do not suppose that every possible need will be filled by these routines so that no others will ever have to be written. In any case, these, like all the other routines we have discussed, will have to be recoded for different makes and models of computers. We hope that these routines will serve most needs, particularly in the printing of intermediate results not intended for publication.